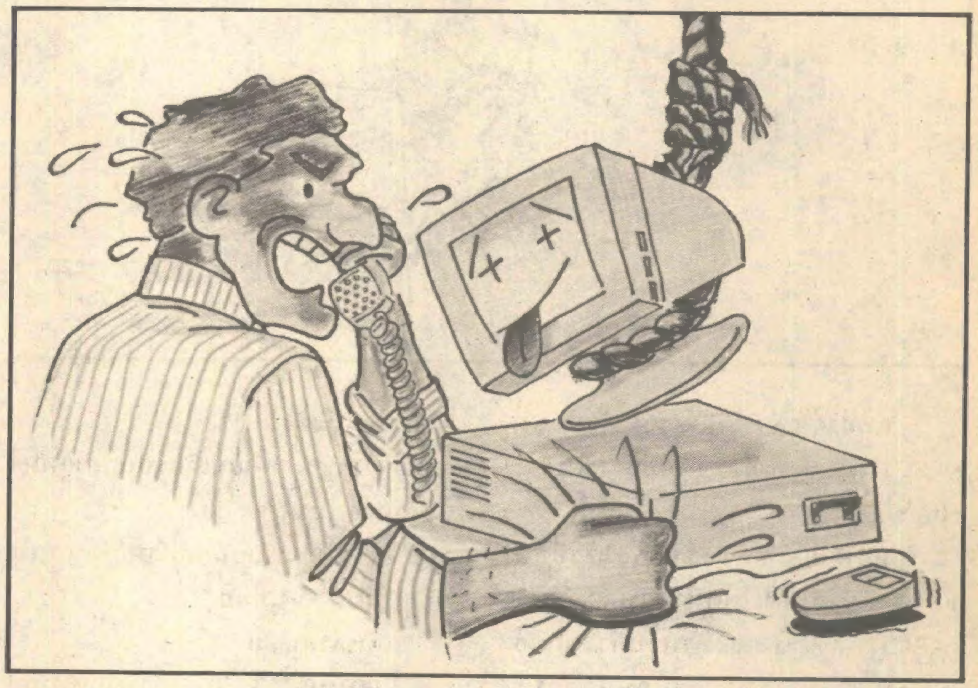
SOFT MAIL

เทคนิคการสร้างเมนูแบบ Pull-Down ในโหมดกราฟิก

อธิบายการทำงาน ของรูทีนต่าง ๆ ในการสร้างเมนูแบบพลูดาวน์ ในโหมดกราฟิก



QuickBASIC เป็นโปรแกรมที่ได้รับการพัฒนาโดยบริษัทไมโครซอฟท์ ปัจจุบันได้อัพเดตเป็นเวอร์ชัน 4.50 QuickBASIC เป็นภาษาที่สามารถเรียนรู้ได้โดยง่าย และสามารถนำมาประยุกต์ใช้งานได้อย่างดีทีเดียว จากที่ผู้เขียนได้แต่งตำราทางด้าน QuickBASIC ทั้ง 3 เล่ม ซึ่งจัดจำหน่ายโดยบริษัทซีเอ็ดยูเคชั่น ก็ปรากฏว่าได้รับความสนใจจากผู้อ่านและโปรแกรมเมอร์ที่เขียนโปรแกรมโดยภาษาเบสิกมากพอสมควร และได้มีจดหมายมาถามเพิ่มเติมเกี่ยวกับการเขียนโปรแกรมขั้นสูง เช่น การสร้างเมนูในโหมดกราฟิก การสร้างเวิร์ดโปรเซสเซอร์ การอ่านฟอนต์จากเวิร์ดจุฬา การอินเตอร์เฟสกับภาษาแอสเซมบลี การตรวจสอบรายละเอียดฮาร์ดแวร์ ซอร์สโค้ดของโปรแกรม BEQB เวอร์ชัน 1.00 และอื่น ๆ ซึ่งรายละเอียดเหล่านี้ผู้เขียนไม่สามารถที่จะตอบคำถามทางจดหมายได้หมด ดังนั้นผู้เขียนจึงขอเปิดบทความกันวารสารไมโครในหัวข้อ Advanced QuickBASIC 4.50 Programming Technique เพื่อหวังว่าจะสามารถแลกเปลี่ยนความรู้และข้อคิดเห็นเกี่ยวกับ QuickBASIC เพื่อให้ผู้อ่านสามารถได้รับประโยชน์สูงสุดจาก QuickBASIC

สำหรับฉบับแรกนี้ผู้เขียนขอเริ่มด้วย เทคนิคการสร้างเมนูแบบ Pull-Down ในโหมดกราฟิก ซึ่งสำหรับโปรแกรมตัวอย่างต่อไปนี้สามารถทำงานได้กับจอภาพทุกชนิด แต่สำหรับจอโมโนโครมที่ขับโดยการ์ดเฮอร์คิวลิส ผู้อ่านต้องรันโปรแกรม MSHERC.COM เสียก่อน เพื่อให้จอภาพสามารถแสดงผลในโหมดกราฟิกได้ โดยในโปรแกรมตัวอย่างนี้ผู้เขียนขอยืมเมนูจากโปรแกรม BEQB เวอร์ชัน 1.00 ซึ่งเขียนโดย QuickBASIC เวอร์ชัน 4.50 (มีในหนังสือบิตแม็พฟอนต์สำหรับ QuickBASIC) ซึ่งรายละเอียดการทำงานของรูทีนต่าง ๆ มีดังนี้

Alt2ASCII

ชื่อ Alt2ASCII

ประเภท โปรแกรมย่อย

หน้าที่ แปลงคีย์ Alt + Character ให้เป็น Character

การประกาศ DECLARE SUB Alt2ASCII (SKey%, Altkey$)

พารามิเตอร์

Skey% ค่าสแกนโค้ดของคีย์ (Alt + character)

Altkey$ ตัวอักษร (ASCII Code) ที่ตรงกับคีย์ (Alt + character)

การส่งค่ากลับ Altkey$ ตัวอักษร (ASCII Code) ที่ตรงกับคีย์ (Alt + character)

หมายเหตุ สำหรับรายละเอียดของค่าของสแกนโค้ดสำหรับตัวอักษร (ASCII) ต่าง ๆ และสำหรับการกดคีย์ (Shift, Ctrl, Alt) ร่วมกับตัวอักษร (ASCII) ใด ๆ ผู้อ่านสามารถดูได้จากภาคผนวก ค ของหนังสือการใช้โปรแกรม QuickBASIC อีกครั้ง สำหรับตัวอย่างของค่าที่ส่งกลับโดยส่วนรูทีนย่อย ดังนี้ สมมติว่าผู้อ่านกดคีย์ Alt+Q ซึ่งค่าสแกนโค้ดของคีย์นี้มีค่าเป็น 16 ดังนั้นรูทีนย่อยนี้จะส่งค่ากลับโดยผ่านทางตัวแปร Altkey$ ซึ่งมีค่าเป็นแอสกี 81 หรือ Q นั่นเอง

BarMenu

ชื่อ BarMenu

ประเภท โปรแกรมย่อย

หน้าที่ Pulldown Menu Driver

การประกาศ DECLARE SUB BarMenu (HPtr%, VPtr%)

พารามิเตอร์

HPtr% พอยน์เตอร์บอกตำแหน่งปัจจุบันของเมนูหลัก

VPtr% พอยน์เตอร์บอกตำแหน่งปัจจุบันของเมนูย่อย

การส่งค่ากลับ

HPtr% = 0 ถ้าหากเมนูถูกยกเลิกโดยที่ผู้อ่านกดคีย์ Cancel หรือโดยการปิดเมนูโดยกดคีย์ Alt หรือมี HPtr% มากกว่าหรือเท่ากับ 1 ตำแหน่งปัจจุบันของเมนูหลัก

VPtr% = 0 ถ้าหากเมนูถูกยกเลิกโดยที่ผู้อ่านกดคีย์ Cancel หรือโดยการปิดเมนูโดยกดคีย์ Alt (ถ้าหาก HPtr% = 0 ค่าของ VPtr% จะไม่มีความหมาย) หรือถ้า VPtr% มากกว่าหรือเท่ากับ 1 ตำแหน่งปัจจุบันของเมนูย่อย

หมายเหตุ สำหรับรูทีนย่อยนี้เป็นหัวใจของเทคนิคการทำเมนูแบบพลูดาวน์ ดังนั้นถ้าหากผู้อ่านต้องการจะแก้ไขวิธีการแสดงผลเมนูแบบพลูดาวน์ เช่น ตำแหน่งเริ่มต้นในการแสดงเมนู การสร้างกรอบสี่เหลี่ยมรอบเมนู หรืออื่น ๆ คุณจะต้องทำการแก้ไขรายละเอียดการแสดงผลในส่วนนี้ และเนื่องจากรูทีนย่อยนี้ไม่มีการตรวจสอบถึงขอบเขตของการแสดงเมนู ดังนั้นในการใช้งานรูทีนย่อยนี้ผู้อ่านจะต้องทำการตรวจสอบรายละเอียด ดังนี้

1. เมนูหลักเริ่มต้นแสดงที่ตำแหน่ง (2, 2)

2. จำนวนรายการของเมนูหลักโดยที่ผลรวมของชื่อ (Caption) ของเมนูหลักจะต้องมีจำนวนตัวอักษรไม่เกินจำนวนตัวอักษรต่อคอลัมน์-3 เนื่องจากเมนูหลักเริ่มต้นแสดงที่คอลัมน์ที่ 2 และสิ้นสุดที่คอลัมน์ก่อนสุดท้าย และแสดงช่องว่าง 1 ช่องก่อนแสดงเมนูหลักแรก

3. จำนวนรายการของเมนูย่อยโดยที่จำนวนรายการของเมนูย่อย (Pull-down Menu) จะต้องมีค่าไม่เกินจำนวนบรรทัดต่อหน้า-4 เนื่องจากเมนูหลักแสดงผลที่บรรทัดที่ 2 และมีกรอบสี่เหลี่ยมล้อมรอบเมนูย่อย

4. จำนวนตัวอักษรของเมนูย่อยจะต้องมีจำนวนตัวอักษรไม่เกินจำนวนตัวอักษรต่อคอลัมน์-ตำแหน่งเริ่มต้นที่แสดงเมนูหลัก-1 เนื่องจากมีกรอบสี่เหลี่ยมล้อมรอบเมนูย่อย

5. ห้ามมีตัวอักษร Shortcut สำหรับเมนูย่อยที่เหมือนกัน (แต่ละเมนูหลักเท่านั้น) เนื่องจากผู้เขียนกำหนดให้ใช้ตัวอักษร & (ASCII 38) นำหน้าตัวอักษรที่ต้องการให้เป็นตัวอักษร Shortcut ดังนั้นถ้าหากในเมนูย่อยหนึ่ง ๆ มีตัวอักษร Shortcut ที่ซ้ำกันเฉพาะตัวแรกเท่านั้นที่สามารถถูกตรวจจับได้โดยโปรแกรม และในการกำหนดตัวอักษร Shortcut ผู้อ่านควรกำหนดเฉพาะตัวอักษร หรือตัวเลขเท่า

นั้น เพราะตัวอักษรเครื่องหมายบางตัวไม่มีค่าสแกนโค้ด เช่น [, {,], } เป็นต้น

6. การกำหนด Caption ของเมนูย่อยควรมีช่องว่างหน้า Caption อย่างน้อย 1 ช่อง เผื่อไว้สำหรับรายการใดที่เป็นเมนูแบบปิด-เปิด (Toggle) ก็จะสามารถแสดงเครื่องหมายถูก (ASCII 251) ในกรณีที่รายการนั้น ๆ เปิด (On)

ByteSize &

ชื่อ ByteSize&

ประเภท ฟังก์ชัน

หน้าที่ คำนวณขนาดของอะเรย์สำหรับจัดเก็บหน้าต่างสี่เหลี่ยม

การประกาศ DECLARE FUNCTION ByteSize& (XG1%, YG1%, XG2%, YG2%)

พารามิเตอร์

XG1% ตำแหน่ง X (pixel) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

YG1% ตำแหน่ง Y (pixel) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

XG2% ตำแหน่ง X (pixel) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

YG2% ตำแหน่ง Y (pixel) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

การส่งค่ากลับ เลขจำนวนเต็มชนิด Long Integer บอกขนาดของอะเรย์ชนิด Double Precision Real สำหรับจัดเก็บกราฟิกตามขนาดสี่เหลี่ยมที่กำหนด

หมายเหตุ สูตรในการหาขนาดของอะเรย์สำหรับจัดเก็บพื้นที่สี่เหลี่ยมในโหมดกราฟิก มีดังนี้

```
ByteSize = (4 + INT(((x2 - x1 + 1)*BPP + 7)/8)
*Planes*(y2 - y1 + 1)) \ AllocVarSize + 1
```

โดยที่

ByteSize = ขนาดของอะเรย์

x1, y1 = ตำแหน่ง (x, y) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

x2, y2 = ตำแหน่ง (x, y) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

BPP = Bits-Per-Pixel-Per-Plane (ขึ้นกับโหมดกราฟิก)

Planes = Graphic Planes (ขึ้นกับโหมดกราฟิก)

AllocVarSize = ขนาดหน่วยความจำสำหรับตัวแปรแต่ละชนิด

สำหรับรายละเอียดของ BPP, Planes และ AllocVarSize ให้หาดูได้จากหัวข้อ 8.8 การทำภาพเคลื่อนที่ของหนังสือการใช้โปรแกรม QuickBASIC อีกครั้ง

CalcTxt2Grp

ชื่อ CalcTxt2Grp

ประเภท โปรแกรมย่อย

หน้าที่ แปลงตำแหน่งของหน้าต่าง (แถว และ คอลัมน์) ให้เป็นตำแหน่งของจุด (X pixel, Y pixel)

การประกาศ DECLARE SUB CalcTxt2Grp (X1%, Y1%, X2%, Y2%, XG1%, YG1%, XG2%, YG2%)

พารามิเตอร์

X1% ตำแหน่ง X (Cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

Y1% ตำแหน่ง Y (Cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

X2% ตำแหน่ง X (Cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

Y2% ตำแหน่ง Y (Cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

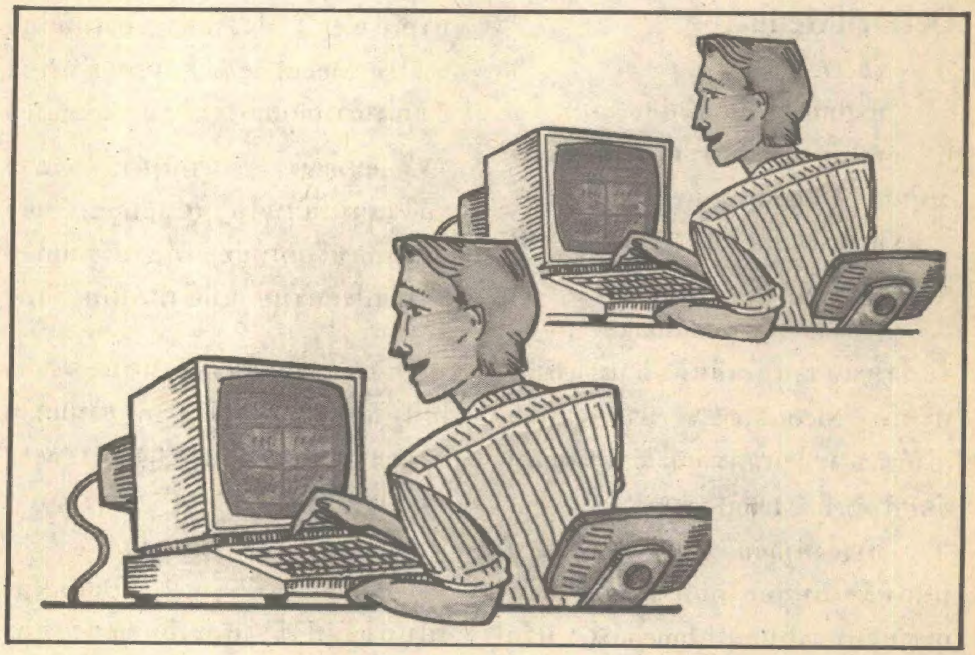
XG1% ตำแหน่ง X (pixel) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

YG1% ตำแหน่ง Y (pixel) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

XG2% ตำแหน่ง X (pixel) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

YG2% ตำแหน่ง Y (pixel) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

การส่งค่ากลับ ตำแหน่งมุม (บนซ้าย และ ล่างขวา) ของหน้าต่าง



สี่เหลี่ยม โดยส่งกลับผ่านทางตัวแปร XG1%, YG1%, XG2% และ YG2%

หมายเหตุ สำหรับสูตรในการคำนวณตำแหน่งจุดของหน้าต่าง (pixel) จากตำแหน่งตัวอักษร (cell) สามารถทำได้ดังนี้

```
XG1% = (X1%-1) * CChr
YG1% = (Y1%-1) * RChr
XG2% = (X2% * CChr)-1
YG2% = (Y2% * RChr)-1
```

โดยที่

CChr = ความกว้างของตัวอักษรในโหมดกราฟิก มีหน่วยเป็นพิกเซล

RChr = ความสูงของตัวอักษรในโหมดกราฟิก มีหน่วยเป็นพิกเซล

สำหรับรายละเอียดของความกว้างและความสูงของตัวอักษรในโหมดกราฟิก ผู้อ่านสามารถหาดูได้จากหัวข้อ 8.1 การ์ดและจอภาพสำหรับการทำงานในโหมดกราฟิก ของหนังสือการใช้โปรแกรม QuickBASIC อีกครั้ง

DefinedMenu

ชื่อ DefinedMenu

ประเภท โปรแกรมย่อย

หน้าที่ กำหนดรายละเอียดของเมนูแบบ Pulldown

การประกาศ DECLARE SUB DefinedMenu (MenuFlag%)

พารามิเตอร์ MenuFlag% = 0 จะกำหนดรายละเอียดของเมนูหลัก หรือถ้า MenuFlag% มากกว่าหรือเท่ากับ 1 จะกำหนดรายละเอียดของเมนูย่อยลำดับที่ MenuFlag%

การส่งค่ากลับ รายการของเมนูหลักหรือเมนูย่อยโดยผ่านทางตัวแปรอะเรย์แบบครอบคลุม hmenu$( ) หรือ

pmenu$( ) โดยขึ้นกับค่าของ MenuFlag% ดังนี้

ถ้าหาก MenuFlag% = 0 ส่งรายการของเมนูหลักกลับโดยผ่านของตัวแปรอะเรย์แบบครอบคลุม hmenu$( )

ถ้าหาก MenuFlag% มากกว่าหรือเท่ากับ 1 ส่งรายการของเมนูย่อยลำดับที่ MenuFag% กลับโดยผ่านทางตัวแปรอะเรย์แบบครอบคลุม pmenu$( )

หมายเหตุ ถ้าหากผู้อ่านต้องการแก้ไขส่วนที่เป็นชื่อ (Caption) ของเมนูหลักหรือเมนูย่อย ก็สามารถแก้ไขได้ในโปรแกรมย่อยนี้ซึ่งมีลำดับ ดังนี้

1. การแก้ไขจำนวนสมาชิกของเมนูหลักหรือเมนูย่อย ให้ผู้อ่านแก้ไขค่าของ UPPER BOUND LIMIT ของประโยค REDIM ซึ่งใช้ในการกำหนดจำนวนสมาชิกของเมนู

2. การแก้ไขชื่อ (Caption) ของเมนูก็ให้แก้ไขรายละเอียดของแต่ละสมาชิกของอะเรย์ (hmenu$( ) หรือ pmenu$( ) ได้ทันที และให้ผู้อ่านใช้ตัวอักษร & (Ampersand) นำหน้าตัวอักษรที่ต้องการให้เป็น ShortCut และในการกำหนดเมนูย่อย ผู้อ่านควรจะเว้นช่องว่างหน้าข้อความอย่างน้อย 1 ช่อง เพื่อใช้ในการแสดงเครื่องหมายถูก (ASCII 251) สำหรับเมนูแบบท็อกเกิล

3. เนื่องจากในที่นี้ผู้เขียนกำหนดให้จำนวนตัวอักษรที่ใช้ในการแสดงผลเมนูย่อยแต่ละกลุ่มจะถูกกำหนดโดยความยาวของเมนูย่อยรายการแรก ดังนั้น ถ้าหากผู้อ่านต้องการให้แสดงผลเมนูย่อยให้มีความกว้างใด ๆ ก็ให้กำหนดที่จำนวนตัวอักษรของเมนูย่อยรายการแรกเท่านั้น

4. การกำหนดเส้นแบ่งกลุ่มของรายการของเมนูย่อย สามารถทำได้โดยการใส่เครื่องหมายลบ - (ASCII 45) เพียงอย่างเดียวในรายการของอะเรย์สำหรับการเลือกรายการของเมนูย่อยโดยการเลื่อนขึ้นลง โปรแกรมจะข้ามไปยังเมนูย่อยถัดไปถ้าหากพบเส้นแบ่งกลุ่มนี้

5. ในการกำหนดเมนูย่อยให้เป็นเมนูแบบเปิด-ปิด (Toggle Menu) ผู้อ่านทำได้โดยการกำหนดตัวแปรในส่วนหลักของโปรแกรม และกำหนดค่าของตัวแปรเป็น TRUE (-1) หรือ FALSE (0) โดยถ้าหากมีค่าเป็น TRUE เมนูก็จะแสดงโดยมีเครื่องหมายถูกข้างหน้า และ ถ้าหากมีค่าเป็น FALSE เมนูก็จะแสดงโดยไม่มีเครื่องหมายถูกข้างหน้า ดังเช่น เมนูย่อย Shift Mode, Bold และ Italic ในเมนูหลัก Transform ซึ่งถูกกำหนดให้เป็นเมนูแบบเปิด-ปิด โดยผ่านทางตัวแปร ShiftFlag%, BoldFlag% และ ItalicFlag% ตามลำดับ เป็นต้น

ตัวแปรค่าคงที่	เส้นเดี่ยว	เส้นคู่
UPPERLEFT	218	201
UPPERRIGHT	191	187
LOWERLEFT	192	200
LOWERRIGHT	217	188
HORIZONTAL	196	205
VERTICAL	179	186

ตารางที่ 1
แสดงการแก้ไขเส้น
แสดงกรอบสี่เหลี่ยม
จากเส้นเดี่ยว
เป็นเส้นคู่

DrawBox

ชื่อ DrawBox

ประเภท โปรแกรมย่อย

หน้าที่ แสดงกรอบสี่เหลี่ยมในพื้นที่ที่กำหนด

การประกาศ DECLARE SUB DrawBox (X1%, Y1%, X2%, Y2%)

พารามิเตอร์

X1% ตำแหน่ง X (Column) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

Y1% ตำแหน่ง Y (Row) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

X2% ตำแหน่ง X (Column) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

Y2% ตำแหน่ง Y (Row) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

การส่งค่ากลับ ไม่มี

หมายเหตุ ถ้าหากผู้อ่านต้องการแก้ไขเส้นที่ใช้แสดงกรอบสี่เหลี่ยมจากเส้นเดี่ยวเป็นเส้นคู่ ก็สามารถแก้ไขค่าของค่าของที่ชนิดท้องที่ (Local Constant) ดังตารางที่ 1

FindBarMenuPos%

ชื่อ FindBarMenuPos%

ประเภท ฟังก์ชัน

หน้าที่ ค้นหาตำแหน่งคอลัมน์เริ่มต้นในการแสดงเมนูหลักลำดับที่ HPtr%

การประกาศ DECLARE FUNCTION FindBarMenuPos% (HPtr%, items$( ))

พารามิเตอร์

HPtr% พอยน์เตอร์บอกตำแหน่งปัจจุบันของเมนูหลัก

item$( ) รายการทั้งหมดของเมนูหลัก

การส่งค่ากลับ เลขจำนวนเต็ม (integer) แสดงตำแหน่งคอลัมน์เริ่มต้นของเมนูลำดับที่กำหนดโดย HPtr%

หมายเหตุ เนื่องจากตัวอักษร & (Ampersand) ผู้เขียนใช้ในการกำหนดตัวอักษร Shortcut ดังนั้นถ้าหากมีตัวอักษรดังกล่าวในรายการของเมนูหลักมากกว่าหนึ่งตัวอักษร & ตัวต่อ ๆ ไปจะถือว่าเป็นส่วนหนึ่งของชื่อ (Caption) ของเมนูหลัก

GetBox

ชื่อ GetBox

ประเภท โปรแกรมย่อย

หน้าที่ จัดเก็บหน้าต่างจอภาพในพื้นที่สี่เหลี่ยมที่กำหนดลงในตัวแปรอะเรย์

การประกาศ DECLARE SUB GetBox (X1%, Y1%, X2%, Y2%, GrpBox#( ))

พารามิเตอร์

X1% ตำแหน่งคอลัมน์ X (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

Y1% ตำแหน่งแถว Y (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

X2% ตำแหน่งคอลัมน์ X (cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

Y2% ตำแหน่งแถว Y (cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

GrpBox#( ) อะเรย์ชนิดเลขจำนวนจริง Double-Precision Real สำหรับจัดเก็บหน้าต่าง

การส่งค่ากลับ รายละเอียดของหน้าต่างที่ถูกจัดเก็บจะถูกส่งกลับผ่านทางตัวแปรอะเรย์ GrpBox#( )

หมายเหตุ สำหรับตำแหน่งมุมทั้งสองของหน้าต่างที่ส่งให้กับโปรแกรมย่อยส่วนนี้จะถูกแปลงให้เป็นตำแหน่งของจุดในโหมดกราฟิกโดยโปรแกรมย่อย CalcTxt2Grp แล้วจึงทำการจองที่ให้กับตัวแปรอะเรย์ GrpBox#( ) แล้วจึงจัดเก็บหน้าต่างสี่เหลี่ยมโดยประโยค GET

GetEvent%

ชื่อ GetEvent%

ประเภท ฟังก์ชัน

หน้าที่ รับตัวอักษรจากการกดคีย์บอร์ดของผู้อ่านโดยไม่มีการคอย คล้ายฟังก์ชัน INKEY$ ของ QuickBASIC

การประกาศ DECLARE FUNCTION GetEvent% (SKey%, Altkey$)

พารามิเตอร์

SKey% ค่าสแกนโค้ดของคีย์ที่ถูกกดโดยผ่านทางคีย์บอร์ด

Altkey$ ตัวอักษร (ASCII Code) ที่ตรงกับคีย์ที่กด

การส่งค่ากลับ Altkey$ ตัวอักษร (ASCII Code) ที่ตรงกับคีย์ที่กด

หมายเหตุ ในการกดคีย์ของผู้อ่านในบางครั้งมีทั้งการกดคีย์แบบร่วมกับ

คีย์พิเศษ เช่น Shift, Ctrl หรือ Alt เป็นต้น ดังนั้นจำนวนไบต์ที่ถูกสร้างขึ้นมาแทนคีย์เหล่านั้นจึงมีขนาดไม่เท่ากัน ผู้เขียนจึงกำหนดประเภทของคีย์ที่กดโดยผ่านทางตัวแปร SKey% ดังนี้

Skey% = 2 ถ้าหากผู้อ่านมีการกดคีย์แบบร่วมกับคีย์พิเศษ เช่น Shift, Ctrl หรือ Alt และ จะส่งกลับ ASCII ของตัวอักษรนั้น ๆ กลับผ่านทางตัวแปร Altkey$

SKey% = 1 ถ้าหากผู้อ่านกดคีย์ ASCII อักษรทั่วไป เช่น A, B, C, 1, 2 เป็นต้น

SKey% = 0 ถ้าหากไม่มีการกดคีย์ใด ๆ เลย

GetEventWait%

ชื่อ GetEventWait%

ประเภท ฟังก์ชัน

หน้าที่ รับตัวอักษรจากการกดคีย์บอร์ดของผู้อ่านโดยการรอจนกว่าผู้อ่านจะกดคีย์ใด ๆ

การประกาศ DECLARE FUNCTION GetEventWait% (SKey%, Altkey$)

พารามิเตอร์

Skey% ถ้าสแกนโค้ดของคีย์ที่ถูกกดโดยผ่านทางคีย์บอร์ด

Altkey$ ตัวอักษร (ASCII Code) ที่ตรงกับคีย์ที่กด

การส่งค่ากลับ Altkey$ ตัวอักษร (ASCII Code) ที่ตรงกับคีย์ที่กด

หมายเหตุ ในการกดคีย์ของผู้อ่านในบางครั้งมีทั้งการกดคีย์แบบร่วมกับคีย์พิเศษ เช่น Shift, Ctrl หรือ Alt เป็นต้น ดังนั้นจำนวนไบต์ที่ถูกสร้างขึ้นมาแทนคีย์เหล่านั้นจึงมีขนาดไม่เท่ากัน ผู้เขียนจึงกำหนดประเภทของคีย์ที่กดโดยผ่านทางตัวแปร SKey% ดังนี้

SKey% = 2 ถ้าหากผู้อ่านมีการกดคีย์แบบร่วมกับคีย์พิเศษ เช่น Shift, Ctrl หรือ Alt และ จะส่งกลับแอสกีของตัวอักษรนั้น ๆ กลับผ่านทางตัวแปร Altkey$

SKey% = 1 ถ้าหากผู้อ่านกดคีย์ ASCII อักษรทั่วไป เช่น A, B, C, 1, 2 เป็นต้น

SKey% = 0 ถ้าหากไม่มีการกดคีย์ใด ๆ เลย

ส่วนในกรณีที่ผู้อ่านกดคีย์ Alt รูทีนย่อย GetEvent% และ GetEventWait% จะส่งกลับค่าของตัวแปร SKey% = FALSE

InverseBox

ชื่อ InverseBox

ประเภท โปรแกรมย่อย

หน้าที่ กลับสีระหว่างตัวอักษรและ สีพื้นของพื้นที่สี่เหลี่ยมที่กำหนด

การประกาศ DECLARE SUB InverseBox (X1%, Y1%, X2%, Y2%)

พารามิเตอร์

X1% ตำแหน่งคอลัมน์ X (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

Y1% ตำแหน่งแถว Y (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

X2% ตำแหน่งคอลัมน์ X (cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

Y2% ตำแหน่งแถว Y (cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

การส่งค่ากลับ ไม่มี

หมายเหตุ ในการกลับสีกรอบสี่เหลี่ยม (Inverse) สำหรับในโปรแกรมนี้ผู้เขียนใช้วิธีการเก็บพื้นที่สี่เหลี่ยมจึงแสดงผลกลับตรงตำแหน่งเดิมโดยคำสั่ง PUT โดยมีพารามิเตอร์ PRESET ดังนั้นผู้อ่านจึงไม่สามารถที่จะกำหนดสีให้การทำอินเวอร์สได้

Keypressed

ชื่อ Keypressed

ประเภท โปรแกรมย่อย

หน้าที่ รอการกดคีย์ใด ๆ จากผู้อ่าน

การประกาศ DECLARE SUB Keypressed ( )

พารามิเตอร์ ไม่มี

การส่งค่ากลับ ไม่มี

หมายเหตุ โปรแกรมย่อยนี้มีการทำงานคล้ายกับประโยค SLEEP ของ QuickBASIC แต่โปรแกรมย่อยนี้จะไม่มีการตรวจสอบเหตุการณ์ (Event) ต่าง ๆ เช่น เวลา, เออเรอ เป็นต้น

OutTextInGraphic

ชื่อ OutTextInGraphic

ประเภท โปรแกรมย่อย

หน้าที่ แสดงผลข้อความตรงตำแหน่งที่กำหนดในโหมดกราฟิก

การประกาศ DECLARE SUB OutTextInGraphic (XPos%, YPos%, TextLen%, InverseFlag%, msg$)

พารามิเตอร์

XPos% ตำแหน่งคอลัมน์ X (cell) ที่เริ่มต้นแสดงข้อความ

YPos% ตำแหน่งแถว Y (cell) ที่เริ่มต้นแสดงข้อความ

TextLen% = 0 ให้แสดงข้อความตามความยาวของข้อความ msg$ หรือถ้า TextLen% มากกว่า 0 ให้แสดงข้อความโดยมีความยาวเท่ากับ TextLen% และถ้าหากข้อความใน msg$ มีความยาวน้อยกว่า TextLen% ก็ให้ต่อท้ายข้อความที่แสดงผลด้วยช่องว่าง (ASCII 32)

InverseFlag% = 0 (FALSE) ให้แสดงผลข้อความตามปกติ หรือถ้า InverseFlag% มากกว่าหรือน้อยกว่า 0 (TRUE) ให้แสดงผลข้อความแบบกลับสี (Inverse)

การส่งค่ากลับ ไม่มี

หมายเหตุ โปรแกรมย่อยนี้ถูกสร้างเพื่อแสดงผลข้อความในโหมดกราฟิก โดยมีการทำงานคล้ายกับประโยค PRINT โดยมีคุณสมบัติ ดังนี้

1. สามารถแสดงผลแบบกลับสี (Inverse) ได้ โดยการกำหนด InverseFlag% = TRUE

2. สามารถระบุตำแหน่งเริ่มต้นการแสดงผลข้อความได้ โดยการกำหนดค่าลงใน XPos%, YPos%

3. ถ้าหากต้องการแสดงผลข้อความตามความหมายที่กำหนดใน msg$ ก็ให้กำหนด TextLen% = 0 แต่ถ้าหากต้องการแสดงผลข้อความใน msg$ ให้มีความยาวใด ๆ ก็ให้กำหนดความยาวของข้อความที่ต้องการแสดงผลใน TextLen%

4. เนื่องจากโปรแกรมย่อยนี้ไม่มีการตรวจจับการสิ้นสุดบรรทัด ดังนั้นถ้าหากข้อความที่แสดงผลมีความยาวเกินกว่าที่จะแสดงในบรรทัดเดียวกันได้ ข้อความส่วนที่เหลือก็จะถูกตัดทิ้ง

บิตที่	ค่าประจำบิต (2^n)	คีย์พิเศษ
0	1	Right Shift (Pressed)
1	2	Left Shift (Pressed)
2	4	Control (Pressed)
3	8	Alternate (Pressed)
4	16	Scroll Lock (On)
5	32	NumLock (On)
6	64	Caps Lock (On)
7	128	Insert (On)

ตารางที่ 2
แสดงคีย์พิเศษที่ต้องการรับให้กับรีจิสเตอร์ AL

PutBox

ชื่อ PutBox

ประเภท โปรแกรมย่อย

หน้าที่ จัดเก็บหน้าต่างจอภาพในพื้นที่สี่เหลี่ยมที่กำหนดลงในตัวแปรอะเรย์

การประกาศ DECLARE SUB PutBox (X1%, Y1% GrpBox#( ))

พารามิเตอร์

X1% ตำแหน่งคอลัมน์ X (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

Y1% ตำแหน่งแถว Y (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

GrpBox#( ) อะเรย์ชนิดเลขจำนวนจริง Double-Precision Real สำหรับแสดงผลหน้าต่าง

การส่งค่ากลับ รายละเอียดของหน้าต่างที่จะถูกแสดงผลจะถูกจัดเก็บในตัวแปรอะเรย์ GrpBox#( ) โดยโปรแกรมย่อย GetBox

หมายเหตุ สำหรับตำแหน่งมุมบนซ้ายของหน้าต่างที่ส่งให้กับโปรแกรมย่อยส่วนนี้ จะถูกแปลงให้เป็นตำแหน่งของจุดในโหมดกราฟิกโดยโปรแกรมย่อย CalcTxt2Grp แล้วจึงกำหนดค่านี้ให้กับประโยค PUT เพื่อแสดงผลหน้าต่างตามตำแหน่งที่กำหนด

Qualifier%

ชื่อ Qualifier%

ประเภท ฟังก์ชัน

หน้าที่ รับการกดคีย์พิเศษ เช่น Left Shift, Right Shift, Control, Alternate เป็นต้น

การประกาศ DECLARE FUNCTION Qualifier% (mask%)

พารามิเตอร์ mask% บิตที่ระบุให้กับฟังก์ชัน เพื่อให้รอรับคีย์ที่กำหนด

การส่งค่ากลับ ค่าของบิตตามที่กำหนดใน mask%

หมายเหตุ เนื่องจาก QuickBASIC ไม่มีฟังก์ชันพิเศษที่สามารถให้เราตรวจจับคีย์พิเศษเหล่านี้ได้ ดังนั้นผู้เขียนจึงได้พัฒนาฟังก์ชันขึ้นมา โดยการเรียกใช้ฟังก์ชันหมายเลข 02H อินเตอร์รัพต์ 16H (Get Keyboard flags) ซึ่งจะกำหนดคีย์พิเศษที่ต้องการรับให้กับรีจิสเตอร์ AL (ไบต์ Keyboard flags จะถูกจัดเก็บในพื้นที่ข้อมูลของ ROM BIOS ตำแหน่ง 0000:0417H ซึ่งถูกระบุโดยบิต) ดังในตารางที่ 2

สำหรับโปรแกรม SKEYDEMO.BAS ต่อไปนี้แสดงตัวอย่างการใช้ฟังก์ชัน Qualifier% ในการตรวจสอบการทำงานของคีย์พิเศษ

ลิสติงที่ 1 SKEYDEMO.BAS

```
' Program        : Shift Key Detecter Program          Version 1.00
' Langauge       : Microsoft QuickBASIC                Version 4.50
' Written By     : Sutthisak Phongthanapanich
' Date           : 8 November 1993
' Library        : QB.QLB (QB.LIB)
' CopyRight      : (C)1993  By Sutthisak Phongthanapanich
'

' $DYNAMIC
' $INCLUDE: 'QB.BI'

' Function declaration.
DECLARE FUNCTION Qualifier% (mask%)

CONST RIGHTSHIFT = 1
CONST LEFTSHIFT = 2
CONST CONTROLSHIFT = 4
CONST ALTERNATESHIFT = 8
CONST SCROLLLOCKSHIFT = 16
CONST NUMLOCKSHIFT = 32
CONST CAPSLOCKSHIFT = 64
CONST INSERTSHIFT = 128

CONST BACKGROUND = 0
CONST FOREGROUND = 15

' Set to check for every special keys.
mask% = 255

CLS
PRINT "Special Key Detecter Program          Version 1.00"
PRINT
PRINT
PRINT "Shift Key State :"
PRINT "Right Shift   "
PRINT "Left Shift    "
PRINT "Control       "
PRINT "Alternate     "
PRINT "ScrollLock    "
PRINT "NumLock       "
PRINT "CapsLock      "
PRINT "Insert        "
PRINT
PRINT "Press 'ESC' to quit."
COLOR FOREGROUND, BACKGROUND
DO
   k% = Qualifier%(mask%)
   p$ = INKEY$

   LOCATE 5, 15
   IF k% AND RIGHTSHIFT THEN PRINT "PRESSED " ELSE PRINT "RELEASED"
   LOCATE 6, 15
   IF k% AND LEFTSHIFT THEN PRINT "PRESSED " ELSE PRINT "RELEASED"
   LOCATE 7, 15
   IF k% AND CONTROLSHIFT THEN PRINT "PRESSED " ELSE PRINT "RELEASED"
   LOCATE 8, 15
   IF k% AND ALTERNATESHIFT THEN PRINT "PRESSED " ELSE PRINT "RELEASED"
   LOCATE 9, 15
   IF k% AND SCROLLLOCKSHIFT THEN PRINT "ON " ELSE PRINT "OFF"
   LOCATE 10, 15
   IF k% AND NUMLOCKSHIFT THEN PRINT "ON " ELSE PRINT "OFF"
   LOCATE 11, 15
   IF k% AND CAPSLOCKSHIFT THEN PRINT "ON " ELSE PRINT "OFF"
   LOCATE 12, 15
   IF k% AND INSERTSHIFT THEN PRINT "ON " ELSE PRINT "OFF"

LOOP UNTIL p$ = CHR$(27)        ' ESC to quit.
' End program.
END

'
' FUNCTION:     Returns the status of the special qualifier keys.
' PARAMETERS:   Right Shift      1
'               Left Shift       2
'               Control          4
'               Alternate        8
'               ScrollLock       16
'               NumLock          32
'               CapsLock         64
'               Insert           128
'
FUNCTION Qualifier% (mask%) STATIC

  DIM Reg AS RegType
  Reg.ax = &H200
  INTERRUPT &H16, Reg, Reg
  Qualifier% = Reg.ax AND mask%

END FUNCTION
```

RemoveShortCutSign$

ชื่อ RemoveShortCutSign$

ประเภท ฟังก์ชัน

หน้าที่ ตัดเครื่องหมายแสดงตัวอักษร Shortcut ออกจากข้อความ (menu caption)

การประกาศ DECLARE FUNCTION RemoveShortCutSign$(msg$)

พารามิเตอร์ msg$ ข้อความที่ต้องการแสดงผล

การส่งค่ากลับ ข้อความที่ถูกตัดเครื่องหมายแสดงตัวอักษร ShortCut ออกเรียบร้อยแล้ว

หมายเหตุ เนื่องจากในการแสดงผลตัวอักษร Shortcut ของเมนูแบบพูลดาวน์ในที่นี้ผู้เขียนใช้เครื่องหมาย & (Ampersand) เป็นตัวกำหนด ดังนั้นถ้าหากข้อความที่ส่งให้กับฟังก์ชันนี้มีเครื่องหมายดังกล่าวอยู่ด้วย เครื่องหมายนี้ก็จะถูกตัดออกจากข้อความ ข้อสังเกตถ้าหากในข้อความเดียวกันมีเครื่องหมายนี้มากกว่าหนึ่งตัว เฉพาะตัวแรกที่ถูกค้นพบเท่านั้นที่จะถูกตัดออกจากข้อความ ส่วนเครื่องหมายตัวต่อ ๆ ไปก็จะถือว่าเป็นส่วนหนึ่งของข้อความ

ScrollBox

ชื่อ ScrollBox

ประเภท โปรแกรมย่อย

หน้าที่ เลื่อนบรรทัดหรือลบข้อความในพื้นที่สี่เหลี่ยมที่กำหนด

การประกาศ DECLARESUB ScrollBox (X1%, Y1%, X2%, Y2%, Lines%, Attb%)

พารามิเตอร์

X1% ตำแหน่งคอลัมน์ X (Cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

Y1% ตำแหน่งแถว Y (cell) บนซ้ายของกรอบหน้าต่างสี่เหลี่ยม

X2% ตำแหน่งคอลัมน์ X (cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

Y2% ตำแหน่งแถว Y (cell) ล่างขวาของกรอบหน้าต่างสี่เหลี่ยม

Lines% กำหนดจำนวนบรรทัดที่ต้องการเลื่อนขึ้น หรือ ลง

Attb% แอตทริบิวต์ที่ใช้ระบายพื้นที่ที่ถูกเลื่อนหรือลบ

การส่งค่ากลับ ไม่มี

หมายเหตุ สำหรับรายละเอียดการใช้งานโปรแกรมย่อยนี้ให้ดูได้จากโปรแกรม BIOS.BAS ในหนังสือ QuickBASIC Toolbox ซึ่งแต่งโดยผู้เขียนเอง

SearchShortCut$

ชื่อ SearchShortCut$

ประเภท ฟังก์ชัน

หน้าที่ ค้นหาตัวอักษร Shortcut ทั้งหมดของเมนูย่อยในแต่ละเมนูหลัก

การประกาศ DECLARE FUNCTION SearchShortCut$ (item$( ))

พารามิเตอร์ item$( ) เมนูย่อย ในที่นี้หมายถึง Pmenu$( )) ของเมนูปัจจุบัน

การส่งค่ากลับ ตัวอักษร Shortcut ของแต่ละเมนูย่อย

หมายเหตุ การกำหนดตัวอักษร Shortcut ให้กับแต่ละเมนู (ทั้งเมนูหลักและเมนูย่อย) มีข้อจำกัดดังนี้

1. ทุก ๆ เมนูจะต้องมีตัวอักษร Shortcut เพราะถ้าหากเมนูใดไม่มีตัวอักษร Shortcut ฟังก์ชันนี้จะกำหนดให้เมนูนี้เป็นเมนูเส้นแบ่งรายการ (ASCII 45) ดังนั้นในการเลื่อนไปมาระหว่างเมนู เมนูนี้จะถูกข้ามโดยอัตโนมัติ

2. ตัวอักษรที่ถูกกำหนดให้เป็นตัวอักษร Shortcut ทั้งตัวอักษรพิมพ์เล็กและตัวอักษรพิมพ์ใหญ่จะถูกแปลงให้เป็นตัวอักษรพิมพ์ใหญ่ทั้งหมด (โดยฟังก์ชันนี้)

SetGCType

ชื่อ SetGCtype

ประเภท โปรแกรมย่อย

หน้าที่ กำหนดรายละเอียดของโหมดกราฟิกแต่ละโหมดผ่านทางตัวแปรโครงสร้าง GC

การประกาศ DECLARE SUB SetGCType (GC AS GCType)

พารามิเตอร์ GC ตัวแปรชนิดโครงสร้าง GCtype

การส่งค่ากลับ ผ่านทางตัวแปรชนิดโครงสร้าง GC

หมายเหตุ สำหรับสมาชิกของตัวแปรโครงสร้าง ดังในรูปที่ 1

สำหรับการกำหนดแกนของหน้าจอ (ทิศทางการเพิ่มหรือลดโคออร์ดิเนต)

BMode	กราฟิกโหมด
MaxX	จำนวนจุด (pixel) ในแนวแกน X (ความกว้างของจอภาพ)
MaxY	จำนวนจุด (pixel) ในแนวแกน Y (ความสูงของจอภาพ)
CChr	จำนวนจุด (pixel) ในแนวแกน X สำหรับตัวอักษรแต่ละตัว (ความกว้างของตัวอักษร)
RChr	จำนวนจุด (pixel) ในแนวแกน Y สำหรับตัวอักษรแต่ละตัว (ความสูงของตัวอักษร)
BPP	Bits-Per-Planes
Plane	Graphic Planes
MaxLine	จำนวนบรรทัดต่อหน้า

รูปที่ 1
โครงสร้างสมาชิกของตัวแปร

ลิสติงที่ 2
GMDEMO.BAS

```
' Program      : Graphic Mode Detecter          Version 1.00
' Langauge     : Microsoft QuickBASIC           Version 4.50
' Written By   : Sutthisak Phongthanapanich
' Date         : 8 November 1993
' Library      : None
' CopyRight    : (C)1993  By Sutthisak Phongthanapanich
'

DIM GMode AS INTEGER

' Search graphic mode by error detect method.
ON ERROR GOTO VideoErr
GMode = 12
SCREEN GMode

PRINT "Graphic Mode : "; GMode
PRINT

PRINT "Press any key to quit."
DO
LOOP UNTIL INKEY$ <> ""

' Restore to text mode.
SCREEN 0
END

VideoErr:
  SELECT CASE GMode
  CASE 12: GMode = 11
  CASE 11: GMode = 10
  CASE 10: GMode = 9
  CASE 9: GMode = 3
  CASE ELSE
     SCREEN 0
     PRINT "Graphic mode unavailable."
     PRINT "Program Terminate."
     END
  END SELECT
  RESUME
```

จะกำหนดให้เป็นแบบ SCREEN COORDINATE คือ จุดเริ่มต้น (0,0) อยู่ที่มุมบนซ้ายของจอภาพ และทิศทางของแกน Y มีทิศทางเพิ่มขึ้นโดยชี้จากบนลงล่าง โดยผู้อ่านสามารถหาอ่านรายละเอียดได้จากหัวข้อที่ 8.4 การกำหนดขอบเขตการแสดงผล ในหนังสือการใช้โปรแกรม QuickBASIC

และสำหรับวิธีการตรวจสอบกราฟิกโหมดที่เหมาะสมกับจอภาพ (และการ์ด) นั้น มีอยู่ด้วยกันหลาย ๆ วิธี แต่สำหรับโปรแกรมนี้ผู้เขียนเลือกใช้วิธีการตรวจจับข้อผิดพลาด เพราะจอภาพบางชนิดที่ต่างกันจะไม่สามารถทำงานในบางกราฟิกได้โดยในโปรแกรมตัวอย่าง GMDEMO.BAS จะแสดงการเลือกกราฟิกโหมดที่เหมาะสมกับจอภาพแต่ละชนิด

ส่งท้าย

จากที่กล่าวมาแล้วข้างต้นทั้งหมดเป็นรายละเอียดและตัวอย่างการใช้งานรูทีนต่าง ๆ ของโปรแกรม HMENU.BAS ซึ่งผู้เขียนได้พัฒนาขึ้นมาเพื่อหวังให้เป็นแนวทางในการปรับปรุงให้ดีขึ้นตามความต้องการของผู้อ่านเท่านั้น ดังนั้นหลังจากที่ผู้อ่านได้ศึกษารายละเอียดการทำงานของโปรแกรมอย่างละเอียดแล้ว ผู้อ่านสามารถที่จะทำการปรับปรุงความสามารถของรูทีนการแสดงผลเมนูแบบพลูดาวน์ให้ดียิ่งขึ้นได้ เช่น การเพิ่มคีย์ Shortcut, การแสดงเมนูเป็นสีหรืออื่น ๆ และสำหรับโปรแกรม HMENU.BAS ต่อไปนี้ ก็จะเป็นโปรแกรมตัวอย่างการเรียกใช้รูทีนการสร้างเมนูแบบพลูดาวน์ และสำหรับในฉบับต่อไปผู้เขียนก็จะเสนอเทคนิคการสร้างเมนูของ Popup ในโหมดกราฟิก

ก็ขอให้ผู้อ่านคอยติดตามต่อไปนะครับ

```
' Program        : Pull-Down Menu Driver (Graphics Mode)   Version 1.00
' Langauge       : Microsoft QuickBASIC                    Version 4.50
' Written By     : Sutthisak Phongthanapanich
' Date           : 4 November 1993
' Library        : QB.QLB (QB.LIB)
' CopyRight      : (C)1993  By Sutthisak Phongthanapanich

' $DYNAMIC
' $INCLUDE: 'QB BI'

' Graphics configuration type.
TYPE GCtype
    BMode       AS INTEGER
    MaxX        AS INTEGER
    MaxY        AS INTEGER
    CChr        AS INTEGER
    RChr        AS INTEGER
    BPP         AS INTEGER
    Plane       AS INTEGER
    MaxLine     AS INTEGER
END TYPE

' Global constant declaration.
CONST FALSE = 0, TRUE = NOT FALSE

' Function declaration.
DECLARE FUNCTION ByteSize& (XG1%, YG1%, XG2%, YG2%)
DECLARE FUNCTION GetEvent% (SKey%, Altkey$)
DECLARE FUNCTION GetEventWait% (SKey%, Altkey$)
DECLARE FUNCTION FindBarMenuPos% (HPtr%, item$())
DECLARE FUNCTION RemoveShortCutSign$ (msg$)
DECLARE FUNCTION SearchShortCut$ (item$())
DECLARE FUNCTION Qualifier% (mask%)

' Subprogram declaration.
DECLARE SUB Alt2ASCII (SKey%, Altkey$)
DECLARE SUB BarMenu (HPtr%, VPtr%)
DECLARE SUB CalcTxt2Grp (X1%, Y1%, X2%, Y2%, XG1%, YG1%, XG2%, YG2%)
DECLARE SUB DefinecMenu (MenuFlag%)
DECLARE SUB DrawBox (X1%, Y1%, X2%, Y2%)
DECLARE SUB GetBox (X1%, Y1%, X2%, Y2%, GrpBox#())
DECLARE SUB InverseBox (X1%, Y1%, X2%, Y2%)
DECLARE SUB Keypressed ()
DECLARE SUB PutBox (X1%, Y1%, GrpBox#())
DECLARE SUB OutTextInGraphic (XPos%, YPos%, TextLen%, InverseFlag%, msg$)
DECLARE SUB ScrollBox (X1%, Y1%, X2%, Y2%, Lines%, Attb%)
DECLARE SUB SetGCType (GC AS ANY)

' Defines global menu variables.
DIM SHARED HMENU$(1), pmenu$(1)
' Allocate space for Graphics variable
DIM SHARED GC AS GCtype
' Allocate space for check menu variables.
DIM ClockFlag%, ShiftFlag%, BoldFlag%, ItalicFlag%
DIM HPtr%, VPtr%
' Allocate space for popup window.
DIM PopUpGrp#(1)

' Defined Grobal variables
ClockFlag% = TRUE
ShiftFlag% = TRUE
BoldFlag% = TRUE
ItalicFlag% = TRUE

' Search graphic mode by error detect method.
ON ERROR GOTO VideoErr
GC.BMode = 12
SCREEN GC.BMode
SetGCType GC

' Make Frame and wall paper.
VIEW (GC.CChr, GC.RChr)-(GC.MaxX - GC.CChr, GC.MaxY - GC.RChr)
DrawBox 1, 1, 80, GC.MaxLine
FOR i% = 1 TO GC.MaxX STEP 10
    CIRCLE (GC.MaxX \ 2, GC.MaxY \ 2), i%
NEXT i%
VIEW
UsageMsg$ = "Press 'Alt' to on/off menu, 'Alt+Underline' or 'Underline' to shortcut. "
OutTextInGraphic 2, GC.MaxLine - 1, 78, TRUE, UsageMsg$
DO
    BarMenu HPtr%, VPtr%

    ' Off menu by 'Alt' key.
    IF HPtr% = 0 THEN EXIT DO
    ' Display selected menu.
    MainMenu$ = RemoveShortCutSign$(HMENU$(HPtr%))
    PullDownMenu$ = RTRIM$(RemoveShortCutSign$(pmenu$(VPtr%)))
    IF LEN(MainMenu$) > LEN(PullDownMenu$) THEN
        PopUpWidth% = LEN(MainMenu$)
    ELSE
        PopUpWidth% = LEN(PullDownMenu$)
    END IF
    X1% = (80 - PopUpWidth% - 16) / 2
    Y1% = (GC.MaxLine - 6) / 2
    X2% = X1% + PopUpWidth% + 17
    Y2% = Y1% + 6
    GetBox X1%, Y1%, X2%, Y2%, PopUpGrp#()
    ScrollBox X1%, Y1%, X2%, Y2%, 0, 0
    DrawBox X1%, Y1%, X2%, Y2%
    LOCATE Y1% + 1, X1% + 1: PRINT "User select :";
    LOCATE Y1% + 2, X1% + 1: PRINT "- Main Menu : "; MainMenu$;
    LOCATE Y1% + 3, X1% + 1: PRINT "- Pull Menu : "; PullDownMenu$;
    OutTextInGraphic X1% + 1, Y1% + 5, X2% - X1% - 1, TRUE, " Press any key to go."
    Keypressed
    PutBox X1%, Y1%, PopUpGrp#()

    ' Toggle Menu demonstration.
    SELECT CASE HPtr%
    CASE 5
        IF VPtr% = 8 THEN ClockFlag% = NOT ClockFlag%    ' Clock
    CASE 4
        IF VPtr% = 1 THEN       ' Shift
            ShiftFlag% = NOT ShiftFlag%
        ELSEIF VPtr% = 2 THEN   ' Bold
            BoldFlag% = NOT BoldFlag%
        ELSEIF VPtr% = 3 THEN   ' Italic
            ItalicFlag% = NOT ItalicFlag%
        END IF
    CASE ELSE
    END SELECT
```

ซอฟต์แวร์
เทคนิคการสร้างเมนูแบบ Pull-Down ในโหมดกราฟิก

ลิสติงที่ 3 HMENU.BAS (ต่อ)

```
LOOP UNTIL HPtr% = 1 AND VPtr% = 15    ' Exit menu.

' Restore to text mode.
SCREEN 0
END

VideoErr:
  SELECT CASE GC.BMode
  CASE 12: GC.BMode = 11
  CASE 11: GC.BMode = 10
  CASE 10: GC.BMode = 9
  CASE 9. GC.BMode = 3
  CASE ELSE
    SCREEN 0
    PRINT "Graphic mode unavailable."
    PRINT "Program Terminate."
    END
  END SELECT
  RESUME

' FUNCTION:      Convert alternate key to equivalent letter.
' PARAMETERS:    SKey%      ' Shift key flag%.
'                Altkey$    Keyboard ascii code.
'
SUB Alt2ASCII (SKey%, Altkey$) STATIC

  SKey% = ASC(Altkey$)
  SELECT CASE SKey%
  CASE 16 TO 25: Altkey$ = MID$("QWERTYUIOP", SKey% - 15, 1)
  CASE 30 TO 38: Altkey$ = MID$("ASDFGHJKL", SKey% - 29, 1)
  CASE 44 TO 50: Altkey$ = MID$("ZXCVBNM", SKey% - 43, 1)
  CASE 104 TO 113: Altkey$ = CHR$(SKey% - 103)
  CASE 120 TO 129: Altkey$ = MID$("1234567890", SKey% - 119, 1)
  CASE ELSE: Altkey$ = ""
  END SELECT
  IF Altkey$ = "" THEN SKey% = TRUE ELSE SKey% = FALSE

END SUB

' FUNCTION:      Horizontal bar main menu.
' PARAMETERS:    HPtr%      Horizontal menu pointer.
'                VPtr%      Vertical menu pointer.
'
SUB BarMenu (HPtr%, VPtr%) STATIC

  REDIM PullDownMenuBox#(1)

  ' Get main menu.
  DefinedMenu 0
  hArrNum% = UBOUND(HMENU$)

  OutTextInGraphic 2, 2, 1, TRUE, " "
  X% = 3
  FOR i% = 1 TO hArrNum%
    OutTextInGraphic X%, 2, 0, TRUE, HMENU$(i%)
    X% = X% + LEN(HMENU$(i%)) - 1
  NEXT i%
  OutTextInGraphic X%, 2, 80 - X%, TRUE, ""

  mainsc$ = SearchShortCut$(HMENU$())
  Ans% = FALSE
  HPtr% = 1

  DO
    X% = FindBarMenuPos%(HPtr%, HMENU$())
    OutTextInGraphic X%, 2, 0, FALSE, HMENU$(HPtr%)
    k% = GetEventWait%(SKey%, Altkey$)
    OutTextInGraphic X%, 2, 0, TRUE, HMENU$(HPtr%)
    IF SKey% = TRUE THEN          ' 2nd Byte OK.
      SELECT CASE Altkey$
      CASE "K"   ' LEFT
        IF HPtr% = 1 THEN HPtr% = hArrNum% ELSE HPtr% = HPtr% - 1
      CASE "M"   ' RIGHT
        HPtr% = (HPtr% MOD hArrNum%) + 1
      CASE "P"      ' Down
        GOSUB PullDownMenu
      CASE ""   ' Alt-pressed-released
        HPtr% = 0
      CASE ELSE
      END SELECT
    ELSE        ' 1st Byte OK.
      SELECT CASE ASC(Altkey$)
      CASE 13: GOSUB PullDownMenu
      CASE 27: HPtr% = 0
      CASE ELSE
        i% = INSTR(mainsc$, Altkey$)
        IF i% THEN HPtr% = i%: GOSUB PullDownMenu
      END SELECT
    END IF
  LOOP UNTIL (HPtr% = 0) OR (Ans% = TRUE)
  EXIT SUB

PullDownMenu:
  Ans% = FALSE

  DO
    VPtr% = 1
    X% = FindBarMenuPos%(HPtr%, HMENU$())
    OutTextInGraphic X%, 2, 0, FALSE, HMENU$(HPtr%)
    ' Get pull down menu for each main menu.
    DefinedMenu HPtr%
    sc$ = SearchShortCut$(pmenu$())
    ArrNum% = UBOUND(pmenu$)
    MenuLen% = LEN(pmenu$(1))
    X1% = X%: Y1% = 3
    X2% = X1% + MenuLen% + 1: Y2% = Y1% + ArrNum% + 1

    ' Keep area that used be pull down menu.
    GetBox X1%, Y1%, X2%, Y2%, PullDownMenuBox#()
    ' Draw pull down menu box.
    DrawBox X1%, Y1%, X2%, Y2%
    ' Print menu items.
    FOR i% = 1 TO ArrNum%
      OutTextInGraphic X1% + 1, Y1% + i%, MenuLen%, FALSE, pmenu$(i%)
    NEXT i%
    InverseBox X1%, Y1%, X2%, Y2%
    DO
      OutTextInGraphic X1% + 1, Y1% + VPtr%, MenuLen%, FALSE, pmenu$(VPtr%)
      k% = GetEventWait%(SKey%, Altkey$)
      OutTextInGraphic X1% + 1, Y1% + VPtr%, MenuLen%, TRUE, pmenu$(VPtr%)
      OldHPtr% = HPtr%
```

```
    IF SKey% = TRUE THEN
        SELECT CASE Altkey$
        CASE "K"      ' Left
            IF HPtr% > 1 THEN HPtr% = HPtr% - 1 ELSE HPtr% = hArrNum%
        CASE "M"      ' Right
            HPtr% = (HPtr% MOD hArrNum%) + 1
        CASE "H"      ' Up
            IF VPtr% = 1 THEN
                VPtr% = ArrNum%
            ELSE
                VPtr% = VPtr% - 1
                IF pmenu$(VPtr%) = "-" THEN VPtr% = VPtr% - 1
                IF VPtr% < 1 THEN VPtr% = ArrNum%
            END IF
        CASE "P"      ' Down
            VPtr% = (VPtr% MOD ArrNum%) + 1
            IF pmenu$(VPtr%) = "-" THEN VPtr% = (VPtr% MOD ArrNum%) + 1
        CASE ""       ' Alt-pressed-released
            VPtr% = 0
        CASE ELSE
        END SELECT
    ELSE
        SELECT CASE ASC(Altkey$)
        CASE 13: Ans% = TRUE
        CASE 27: HPtr% = 0: Ans% = TRUE
        CASE ELSE
            i% = INSTR(sc$, Altkey$)
            IF i% THEN VPtr% = i%: Ans% = TRUE
        END SELECT
    END IF
LOOP UNTIL (Ans% = TRUE) OR (OldHPtr% <> HPtr%) OR (VPtr% = 0)
' Restore area that used be pull down menu.
PutBox X1%, Y1%, PullDownMenuBox#()
OutTextInGraphic X%, 2, 0, TRUE, HMENU$(OldHPtr%)
LOOP UNTIL (Ans% = TRUE) OR (VPtr% = 0)
RETURN

END SUB


' FUNCTION:      Calculate size of array for graphics popup window.
' PARAMETERS:  XG1%, YG1%      Upper left X, Y coordinate.
'              XG2%, YG2%      Lower right X, Y coordinate.
'
FUNCTION ByteSize& (XG1%, YG1%, XG2%, YG2%) STATIC

  k& = INT(((XG2% - XG1% + 1) * GC.BPP) + 7&) / 8
  ByteSize& = (4 + (YG2% - YG1% + 1) * k& * GC.Plane) \ 8 + 1
END FUNCTION


' FUNCTION:      Calculate graphic coordinate from text position.
' PARAMETERS:  X1%, Y1%        Upper left X, Y text position.
'              X2%, Y2%        Lower right X, Y text position.
'              XG1%, YG1%      Upper left X, Y graphic coordinate.
'              XG2%, YG2%      Lower right X, Y graphic coordinate.
'
SUB CalcTxt2Grp (X1%, Y1%, X2%, Y2%, XG1%, YG1%, XG2%, YG2%) STATIC

  XG1% = (X1% - 1) * GC.CChr: YG1% = (Y1% - 1) * GC.RChr
  XG2% = X2% * GC.CChr - 1: YG2% = Y2% * GC.RChr - 1

END SUB


' FUNCTION:      Defines menu massages and box location.
' PARAMETERS:  Menuflag%       Menu item number.
'                              0          for Main menu.
'                              1-32767    for pull down menu.
'
SUB DefinedMenu (MenuFlag%) STATIC
SHARED ClockFlag%, ShiftFlag%, BoldFlag%, ItalicFlag%

  SELECT CASE MenuFlag%
  CASE 0              ' Main Menu
    REDIM HMENU$(1 TO 6)
    HMENU$(1) = " &File "
    HMENU$(2) = " &Edit "
    HMENU$(3) = " &View "
    HMENU$(4) = " &Transform "
    HMENU$(5) = " &Run "
    HMENU$(6) = " &Help "
  CASE 1              ' File
    REDIM pmenu$(1 TO 15)
    pmenu$(1) = " &New..."
    pmenu$(2) = " &Open..."
    pmenu$(3) = " &Creat..."
    pmenu$(4) = " &Save"
    pmenu$(5) = " Save &As..."
    pmenu$(6) = " Dri&ve..."
    pmenu$(7) = "-"
    pmenu$(8) = " &Resize..."
    pmenu$(9) = "-"
    pmenu$(10) = " Print Set&up..."
    pmenu$(11) = " &Print..."
    pmenu$(12) = " &List..."
    pmenu$(13) = " &DOS Shell"
    pmenu$(14) = "-"
    pmenu$(15) = " E&xit..."
  CASE 2              ' Edit
    REDIM pmenu$(1 TO 11)
    pmenu$(1) = " Cu&t"
    pmenu$(2) = " C&opy"
    pmenu$(3) = " &Paste"
    pmenu$(4) = " &Delete"
    pmenu$(5) = " &Fill"
    pmenu$(6) = " &Inverse"
    pmenu$(7) = "-"
    pmenu$(8) = " Cop&y To..."
    pmenu$(9) = " Pa&ste From..."
    pmenu$(10) = "-"
    pmenu$(11) = " &Modify header..."
  CASE 3              ' View
    REDIM pmenu$(1 TO 10)
    pmenu$(1) = " &Ascii table...  "
    pmenu$(2) = "-"
    pmenu$(3) = " Go&to Pixel..."
    pmenu$(4) = " Hori&zontal TAB"
    pmenu$(5) = " &Vertical TAB"
    pmenu$(6) = "-"
    pmenu$(7) = " &Column marker"
    pmenu$(8) = " &Row marker"
    pmenu$(9) = "-"
```

ลิสติงที่ 3 HMENU.BAS (ต่อ)

```
    pmenu$(10) = " Refres&h Screen"
  CASE 4                ' Transform
    REDIM pmenu$(1 TO 13)
    pmenu$(1) = " &Shift Mode      "
    pmenu$(2) = " &Bold"
    pmenu$(3) = " &Italic"
    pmenu$(4) = "-"
    pmenu$(5) = " Flip Hori&zontal"
    pmenu$(6) = " Flip &Vertical"
    pmenu$(7) = " Mirror &X"
    pmenu$(8) = " Mirror &Y"
    pmenu$(9) = "-"
    pmenu$(10) = " Insert &Column"
    pmenu$(11) = " Insert &Row"
    pmenu$(12) = " Re&move Column"
    pmenu$(13) = " R&emove Row"
    IF ShiftFlag% THEN MID$(pmenu$(1), 1, 1) = CHR$(251)
    IF BoldFlag% THEN MID$(pmenu$(2), 1, 1) = CHR$(251)
    IF ItalicFlag% THEN MID$(pmenu$(3), 1, 1) = CHR$(251)
  CASE 5                ' Run
    REDIM pmenu$(1 TO 13)
    pmenu$(1) = " &Verify font      "
    pmenu$(2) = " &Make printer font"
    pmenu$(3) = " Make New &Style font"
    pmenu$(4) = "-"
    pmenu$(5) = " Run &Expression..."
    pmenu$(6) = " Run &File..."
    pmenu$(7) = "-"
    pmenu$(8) = " &Clock"
    pmenu$(9) = " &Time..."
    pmenu$(10) = " D&ate..."
    pmenu$(11) = "-"
    pmenu$(12) = " &Information"
    pmenu$(13) = " &DOS command..."
    IF ClockFlag% THEN MID$(pmenu$(8), 1, 1) = CHR$(251)
  CASE 6                ' Help
    REDIM pmenu$(1 TO 6)
    pmenu$(1) = " &Commands
    pmenu$(2) = " &Short-Cut"
    pmenu$(3) = " Copy&Right"
    pmenu$(4) = "-"
    pmenu$(5) = "-"
    pmenu$(6) = " &About Pull Down Menu Driver"
  CASE ELSE             ' Hold for append.
  END SELECT

END SUB

'
' FUNCTION:       Draw rectangular box.
' PARAMETERS:  X1%, Y1%       Upper left X, Y coordinate.
'                    X2%, Y2%       Lower right X, Y coordinate.
'
SUB DrawBox (X1%, Y1%, X2%, Y2%) STATIC

  CONST UPPERLEFT = 218, UPPERRIGHT = 191
  CONST LOWERLEFT = 192, LOWERRIGHT = 217
  CONST HORIZONTAL = 196, VERTICAL = 179

  LOCATE Y1%, X1%: PRINT CHR$(UPPERLEFT); STRING$(X2% - X1% - 1, HORIZONTAL); CHR$(UPPERRIGHT);
  FOR i% = Y1% + 1 TO Y2% - 1
    LOCATE i%, X1%: PRINT CHR$(VERTICAL);
    LOCATE i%, X2%: PRINT CHR$(VERTICAL);
  NEXT i%
  LOCATE Y2%, X1%: PRINT CHR$(LOWERLEFT); STRING$(X2% - X1% - 1, HORIZONTAL);
CHR$(LOWERRIGHT);

  END SUB

  REM $STATIC
  '
  ' FUNCTION:       Find start position of specified horizontal bar menu.
  ' PARAMETERS:  HPtr%          Horizontal pointer.
  '                    item()          Menu items.
  '
  FUNCTION FindBarMenuPos% (HPtr%, item$()) STATIC

    j% = 3
    FOR i% = 1 TO HPtr% - 1
      j% = j% + LEN(item$(i%))
      IF INSTR(item$(i%), "&") THEN j% = j% - 1
    NEXT i%
    FindBarMenuPos% = j%

  END FUNCTION

  REM $DYNAMIC
  '
  ' FUNCTION:       Save graphic screen.
  ' PARAMETERS:  X1%, Y1%       Upper left X, Y coordinate.
  '                    X2%, Y2%       Lower right X, Y coordinate.
  '                    GrpBox#()      Array for graphics popup window.
  '
  SUB GetBox (X1%, Y1%, X2%, Y2%, GrpBox#()) STATIC

    ' Calculate graphic coordinates from text position.
    CalcTxt2Grp X1%, Y1%, X2%, Y2%, XG1%, YG1%, XG2%, YG2%

    REDIM GrpBox#(1 TO ByteSize&(XG1%, YG1%, XG2%, YG2%))

    GET (XG1%, YG1%)-(XG2%, YG2%), GrpBox#

  END SUB

  '
  ' FUNCTION:       Read character from keyboard.
  ' PARAMETERS:  SKey%          Shift key flag%.
  '                    Altkey$         Keyboard ascii code.
  '
  FUNCTION GetEvent% (SKey%, Altkey$) STATIC

    key$ = INKEY$
    SELECT CASE LEN(key$)
    CASE 2: SKey% = TRUE: Altkey$ = RIGHT$(key$, 1)
    CASE 1: SKey% = FALSE: Altkey$ = UCASE$(key$)
    CASE 0: SKey% = FALSE: Altkey$ = CHR$(0)
    END SELECT

    GetEvent% = CVI(key$ + STRING$(2, 0))

  END FUNCTION
```

```
FUNCTION:       Read character from keyboard (wait)
PARAMETERS:     SKey%           Shift key flag%.
                Altkey$         Keyboard ascii code.

FUNCTION GetEventWait% (SKey%, Altkey$) STATIC

  DO
    k% = GetEvent%(SKey%, Altkey$)
    IF Qualifier%(8) THEN
        DO WHILE Qualifier%(8)
          k% = GetEvent(SKey%, Altkey$)
          IF SKey% THEN EXIT DO
        LOOP
        DO WHILE Qualifier%(8)
        LOOP
        Alt2ASCII SKey%, Altkey$
        EXIT DO
    END IF
  LOOP UNTIL k%

  GetEventWait% = k%

END FUNCTION


  FUNCTION      Inverse rectangular box.
  PARAMETERS:   X1%, Y1%        Upper left X, Y coordinate.
                X2%, Y2%        Lower right X, Y coordinate.

SUB InverseBox (X1%, Y1%, X2%, Y2%) STATIC

  REDIM GrpBox#(1)

  CalcTxt2Grp X1%, Y1%, X2%, Y2%, XG1%, YG1%, XG2%, YG2%
  REDIM GrpBox#(1 TO ByteSize&(XG1%, YG1%, XG2%, YG2%))

  GET (XG1%, YG1%)-(XG2%, YG2%), GrpBox#
  PUT (XG1%, YG1%), GrpBox#, PRESET

END SUB


  FUNCTION:     Wait for one key pressed.

SUB Keypressed STATIC

  DO

  LOOP WHILE INKEY$ = ""

END SUB

REM $STATIC

  FUNCTION:     Output string at specify position.
  PARAMETERS:   XPos%           Start X position.
                YPos%           Start Y position.
                TextLen%        Length of string.
                InverseFlag%    Inverse video flag.
                msg$            Strings to output.

SUB OutTextInGraphic (XPos%, YPos%, TextLen%, InverseFlag%, msg$) STATIC

  p$ = msg$
  ShortCutPos% = INSTR(p$, "&")
  IF ShortCutPos% THEN p$ = LEFT$(p$, ShortCutPos% - 1) + MID$(p$, ShortCutPos% + 1)
  IF TextLen% THEN
    p$ = LEFT$(p$ + SPACE$(TextLen%), TextLen%)
  END IF
  IF RTRIM$(p$) = "-" THEN
    LOCATE YPos%, XPos% - 1
    PRINT CHR$(195); STRING$(TextLen%, CHR$(196)); CHR$(180);
    X1% = XPos% - 1: X2% = X1% + TextLen% + 1
  ELSE
    LOCATE YPos%, XPos%: PRINT p$;
     Display Shortcut character as underline character.
    IF ShortCutPos% THEN
        X1% = XPos% + ShortCutPos% - 1
        X2% = XPos% + ShortCutPos% - 1
        CalcTxt2Grp X1%, 0, X2%, YPos%, XG1%, 0, XG2%, YG2%
        LINE (XG1%, YG2% - 1)-(XG2%, YG2% - 1)
    END IF
    X1% = XPos%: X2% = X1% + LEN(p$) - 1
  END IF

   Make Inverse Text.
  IF InverseFlag% THEN InverseBox X1%, YPos%, X2%, YPos%

END SUB

REM $DYNAMIC

  FUNCTION:     Restore graphic screen.
  PARAMETERS:   X1%, Y1%        Upper left X, Y coordinate.
                GrpBox#()       Array for graphics popup window.

SUB PutBox (X1%, Y1%, GrpBox#()) STATIC

  CalcTxt2Grp X1%, Y1%, 0, 0, XG1%, YG1%, 0, 0
  PUT (XG1%, YG1%), GrpBox#, PSET

END SUB


  FUNCTION:     Returns the status of the special qualifier keys.
  PARAMETERS:   Right Shift     1
                Left Shift      2
                Control         4
                Alternate       8
                ScrollLock      16
                NumLock         32
                CapsLock        64
                Insert          128

FUNCTION Qualifier% (mask%) STATIC

  DIM Reg AS RegType
  Reg.ax = &H200
  INTERRUPT &H16, Reg, Reg
  Qualifier% = Reg.ax AND mask%
```

อิสติงที่ 3 HMENU.BAS (ต่อ)

```
END FUNCTION

'
' FUNCTION:      Remove shortcut sign '&' from string.
' PARAMETERS:    msg$       Menu item.
'
FUNCTION RemoveShortCutSign$ (msg$)

  ShortCutPos% = INSTR(msg$, "&")
  IF ShortCutPos% THEN
    RemoveShortCutSign$ = LEFT$(msg$, ShortCutPos% - 1) + MID$(msg$, ShortCutPos%
  ELSE
    RemoveShortCutSign$ = msg$
  END IF
END FUNCTION

'
' FUNCTION:      Scroll window.
' PARAMETERS:    X1%, Y1%       Upper left x, y coordinate.
'                X2%, Y2%       Lower right x, y coordinate.
'                Lines%         Number of line to blank.
'                               >= 0 Scroll up.
'                               < 0  Scroll down.
'                Attb%          Attribute to filled.
'
SUB ScrollBox (X1%, Y1%, X2%, Y2%, Lines%, Attb%) STATIC

  DIM Reg AS RegType

  IF Lines% > 0 THEN
    Reg.ax = &H600 + Lines% MOD 256
  ELSE
    Reg.ax = &H700 + ABS(Lines%) MOD 256
  END IF

  Reg.bx = (Attb% * 256&) AND &HFF00
  Reg.cx = (Y1% - 1) * 256 + X1% - 1
  Reg.dx = (Y2% - 1) * 256 + X2% - 1
  INTERRUPT &H10, Reg, Reg

END SUB

REM $STATIC

' FUNCTION:      Find short-cut key.
' PARAMETERS:    item$()    Menu items.
'
FUNCTION SearchShortCut$ (item$()) STATIC

  p$ = ""
  FOR i% = 1 TO UBOUND(item$)
    k% = INSTR(item$(i%), "&")
    IF k% THEN
      p$ = p$ + MID$(item$(i%), k% + 1, 1)
    ELSE
      p$ = p$ + "-"
    END IF
  NEXT i%
  SearchShortCut$ = UCASE$(p$)

END FUNCTION

REM $DYNAMIC

'
' FUNCTION:      Set graphics environment.
' PARAMETERS:    GC         Graphics structural variable.
'
SUB SetGCType (GC AS GCtype) STATIC

  SELECT CASE GC.BMode
  CASE 3    ' 2-color high-res graphics for Hercules
    GC.MaxX = 719: GC.MaxY = 347
    GC.BPP = 1: GC.Plane = 1
    GC.CChr = 9: GC.RChr = 14: GC.MaxLine = 25
  CASE 9    ' 16 or 4-color very high-res graphics for EGA and VGA
    GC.MaxX = 639: GC.MaxY = 349
    GC.BPP = 1: GC.Plane = 4
    GC.CChr = 8: GC.RChr = 14: GC.MaxLine = 25
  CASE 10   ' 2-color high-res graphics for EGA or VGA monochrome
    GC.MaxX = 639: GC.MaxY = 349
    GC.BPP = 1: GC.Plane = 2
    GC.CChr = 8: GC.RChr = 14: GC.MaxLine = 25
  CASE 11   ' 2-color very high-res graphics for VGA and MCGA
    GC.MaxX = 639: GC.MaxY = 479
    GC.BPP = 1: GC.Plane = 1
    GC.CChr = 8: GC.RChr = 16: GC.MaxLine = 30
  CASE 12   ' 16-color very high-res graphics for VGA
    GC.MaxX = 639: GC.MaxY = 479
    GC.BPP = 1: GC.Plane = 4
    GC.CChr = 8: GC.RChr = 16: GC.MaxLine = 30
  END SELECT

  SCREEN GC.BMode
  WINDOW SCREEN (0, 0)-(GC.MaxX, GC.MaxY)

END SUB
```